Algorithms and Requirements for Measuring Network Bandwidth

Jin Guojun

Distributed Systems Department
Lawrence Berkeley National Laboratory
1 Cyclotron Road, Berkeley, CA 94720
g_jin@lbl.gov

Abstract — **This report unveils new algorithms for actively measuring (not estimating) available bandwidths with very low intrusion, computing cross traffic, thus estimating the physical bandwidth, provides mathematical proof that the algorithms are accurate, and addresses conditions, requirements, and limitations for new and existing algorithms for measuring network bandwidths. The paper also discusses a number of important terminologies and issues for network bandwidth measurement, and introduces a fundamental parameter — Maximum Burst Size that is critical for implementing algorithms based on multiple packets.**

I. INTRODUCTION

Simple network management protocol (SNMP) [1] can provide detailed statistics on network elements (routers and switches) such as physical bandwidth (capacity [6]), utilization, etc. However, getting network status information via this method requires special access privileges, which are not usually available to ordinary users. Furthermore, using SNMP to obtain path statistics requires significant processor time on each network element along the path, and requires bandwidth to transfer data from all elements back to the inquiry host. This method is useful for network engineers to analyze and study network behavior, and many networks use the Multi Router Traffic Grapher (MRTG) to collect and publish the SNMP results from routers. Active measurement may be a better method to meet ordinary users' needs because it does not require router access.

Algorithms for actively measuring network physical and available bandwidths has been researched for many years. Many tools have been developed, and only a few tools have successfully achieved a close estimation of network bandwidths, defined in section III. *Pathload* is designed to estimate available bandwidth [10]. *Pathchar* is designed to estimate physical bandwidth. *Clink*[4] and *pchar*[17] are different implementations of pathchar. Nettimer[5] uses a passive algorithm to measure the path bottleneck capacity, but this algorithm requires that no queueing occurs on any network element after this bottleneck link, thus, it works on very idealistic paths. Other tools, such as *bprobe/cprobe*[2], *ttcp*[14], *iperf*[15], *netperf*[16], *Sprob*[9], *Treno*[18], are intended to measure bandwidth. However, most of these tools actually measure *achievable throughput*[13].

This paper provides an overview of the existing algorithms for these tools, and unveils new algorithms — FAC^2 (Feedback Adaptive Control and Feedback Asymptotic Convergence) and FSE (Fluid Spray Effect) to measure network bandwidths, and uses mathematical inference to prove that FAC^2 can measure bandwidth accurately. Because FAC^2 measures available bandwidth very quickly (in one second), the result reflects the available bandwidth during that time interval. Applications may need results over large intervals such as 10 seconds or 5 minutes, which will require sampling techniques to determine how to use FAC^2 to probe bandwidth. An important benefit of FAC^2 is that applications can use it to quickly and non intrusively monitor the available bandwidth of a path and adapt to changes in the network. This paper emphasizes the mathematical model and theory, critical implementation issues are discussed, but not the details. The rest paper is organized as:

- Describe the network model, and address terms and issues for measurement
- Distinguish and define bandwidth and throughput
- Introduce and analyze algorithms
- Describe the basic requirements and critical issues for implementing algorithms

II. MODELING THE NETWORK

Building a network model is necessary to determine algorithms for measuring bandwidth. This section describes basic probe mechanism for measuring network,

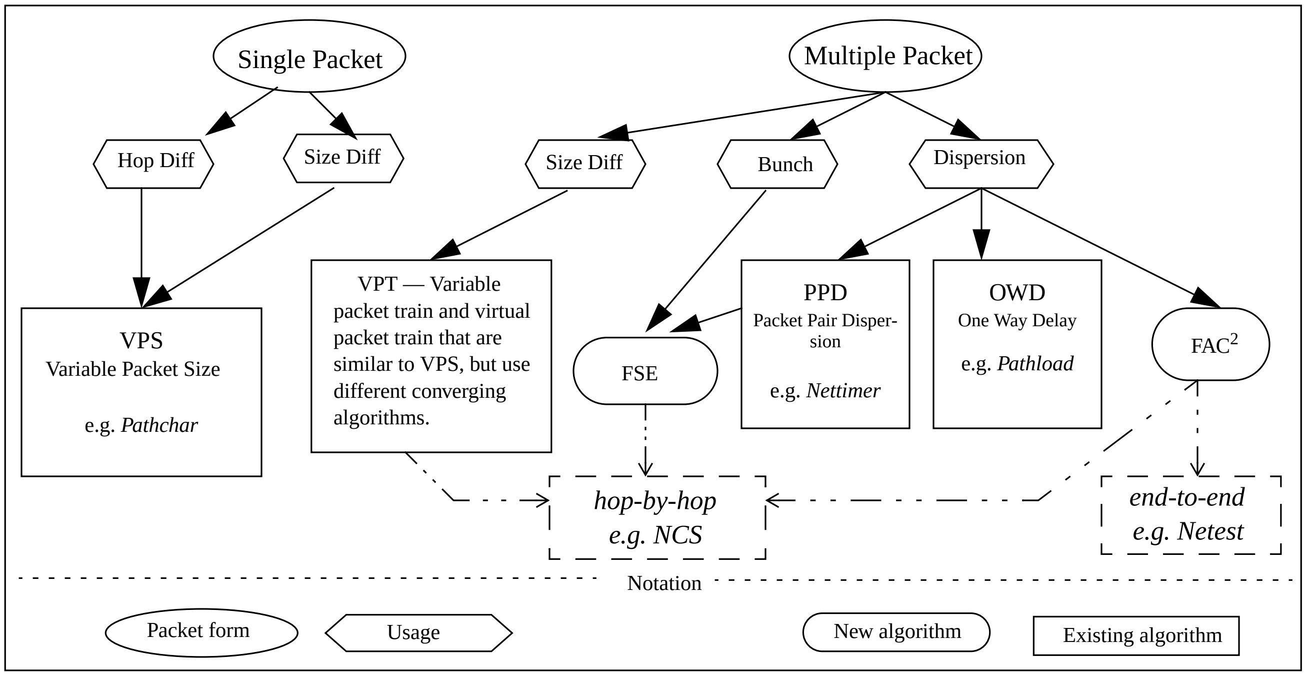


Fig. 1 Using packet to probe network; also relations between algorithms and tools

and the model used to form new algorithms. This section also introduces the concept of the maximum burst size — MBS. A number of terms and abbreviations are defined to address methodology and algorithms for bandwidth measurement. Below is a list of abbreviations and terms used in this paper:

A_{bw}	— available bandwidth
C_p	— capacity of measured path
XT	— cross traffic
PT	— probe traffic or packet train
R_{snd}	— PT sending rate
R_{rcv}	— PT receiving rate
R_{xt}	— cross traffic flow rate

Bit rate — how fast a network interface card (NIC) can put each bit onto the wire. It means the line speed or the transfer rate of single packet.

Train rate (effective bit rate) — the transfer rate of a packet train[8] (multiple packets) traveling through a link.

Fig. 1 characterizes various algorithms used to measure networks, and methods of using packet to probe the network: single packet and packet train (multiple packets). There are several ways such as size differential (SD)[8], hop differential (HD), dispersion, etc., to use these two methods. [8] describes combinations of using SD and HD. [5][6] argue on dispersion technology, and provided partial interest on why it works or not. [11] had some idea on multiple packet technique. Fig. 1 depicts that packet dispersion has great potential in network measurement, and algorithms, FAC^2 and FSE, are based on packet dispersion. Regardless which method to use, both single packet and packet train have requirement issues for measuring high-speed networks. Use of single packet needs a very high-resolution timer, and use of packet train requires higher sending speed than the available bandwidth. The Fig. 1 also shows connections between algorithms and tools.

A. Model

Although networks look like a mesh with stochastic traffic, when analyzing a particular network link or path, all cross traffic at a given network element can be characterized as a single aggregated stream. Thus, we can model a particular network path as two individual traffic streams: cross traffic (XT) and probe traffic (PT), as shown in Fig. 2. This means that the three traffic streams in

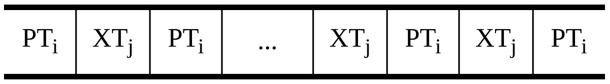


Fig. 2 Simplified network traffic model

Fig. 3 are equivalent (they have the same amount of XT and PT). Using this model helps to solve a number of crucial network problems such as detecting the maximum burst size. In this paper, this model is used with packet

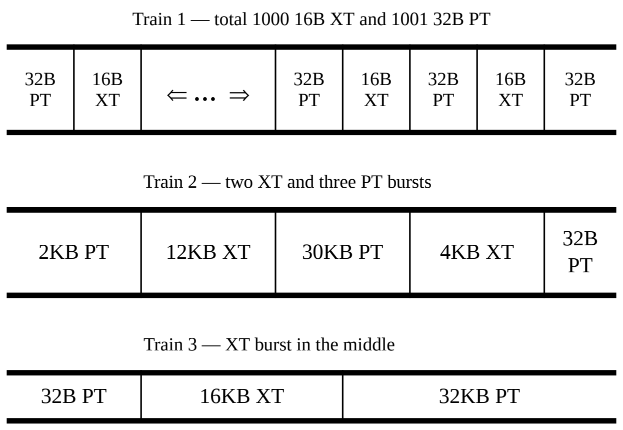

Train 1 — total 1000 16B XT and 1001 32B PT
32B PT
16B XT
⇐ ... ⇒
32B PT
16B XT
32B PT
16B XT
32B PT
Train 2 — two XT and three PT bursts
2KB PT
12KB XT
30KB PT
4KB XT
32B PT
Train 3 — XT burst in the middle
32B PT
16KB XT
32KB PT

Fig. 3 Three equivalent packet trains

train technique to build algorithms to measure network bandwidth.

B. Terminology

A key concept — Maximum Burst Size (MBS) — must be introduced before proceeding to create mathematical models for measuring a network. It is also important to distinguish and define bandwidth and throughput.

> **Maximum Burst Size** (MBS) — the maximum number of bytes that can be sent contiguously from a source host to a destination host across the network during a certain period of time without dropping a packet.

The *maximum burst size* is determined by the queue size of the bottleneck router and by the current network traffic at that router. Note that other factors such as traffic shaping, policing, or QoS may cause such queueing behavior. This can restrict the effective bandwidth when it is small. So, MBS is also called as *effective queue size*. The following example depicts how MBS significantly affects throughput, and this paper discusses how it is important to the bandwidth measurement.

This example illustrates that if MBS is smaller than BDP (bandwidth delay product), it will reduce the effective path bandwidth for normal operation. In this example, the maximum throughput was reduced to one half of the capacity. If applications or operating systems are not aware of this issue, they cannot utilize the available bandwidth and will cause congestion which degrades network performance. MBS is also critical to the packet train method for network measurement. That is, the maximum train length must be less than the MBS; otherwise, the train's tail will be dropped, causing probe failure. How long the train should be configured depends on the variation of the cross traffic.

This example uses a path with 100 ms round trip delay (RTT), while the bottleneck link is 1 Gb/s. Under this circumstance, use of bandwidth delay product (BDP) will set TCP congestion window to

$$0.1s \times 10^9 b/s = 100Mbits = 12.5MBytes$$

If the MBS is 1 MB due to the average cross traffic, the TCP congestion window will be limited to 1 MB because any burst larger than 1 MB will cause packet loss when cross traffic exists, which is almost always true. The maximum TCP throughput then will be

$$\frac{1MB \times 8bits/Byte}{0.1s} = 80Mb/s$$

UDP throughput also depends on the burst size and rate of the traffic. 1 MB effective queue implies that the maximum burst duration (at line speed, 1 Gb/s in this case) for both probe traffic and cross traffic to avoid overflow is:

$$\frac{MBS}{LineSpeed} = \frac{1MB}{1Gb/s} = 8ms$$

because when a burst leaves the previous router, it will travel at link (line) speed. Since we can characterize all cross traffic as one aggregated stream, each stream (aggregated cross traffic and measurement traffic) has minimum 8 ms safety burst period (for router to drain two 1 MB bursts in average according to the effective queue size) to not overflow the smallest queue on this path. Without knowing how to control the burst, UDP throughput will vary depending on how many cross traffic bursts have a duration longer than 8 ms (exceed 1MB in length). The more big bursts of cross traffic, the lower the UDP throughput will be.

Let's choose two burst sizes to illustrate this issue. One UDP stream has a 2MB burst size (twice as big as a safety burst length to cause average 50% packet drop) and 32 ms burst period (twice of the burst duration); the other UDP stream has 0.5MB burst (half the size of the safety burst length to minimize the packet drop) and 8 ms burst period to get the maximum throughput. In the first stream, 31.2 bursts can be sent every second, so the maximum transfer rate should be

$$\frac{2MB \times 8bits/Byte \times 31.2/s}{2} = 249.6Mb/s$$

because of predicted 50% packet drop rate. In the second stream, the maximum throughput should be 500Mb/s since 125 bursts can be transferred in one second. The MBS theory has been examined and verified via SCNM [12] in real cases over different network paths.

In practice, choosing the best burst size can be tricky because MBS is a function of cross traffic, which varies over time. The MBS measured in a short period of time reflects the cross traffic and queueing situation for that moment, and the MBS obtained during a longer period shows the average status during that time interval. None of these MBS can guarantee that using these values at the maximum sending speed will or will not cause the bottleneck router to drop packet in the near future. For example, if a bottleneck router has four incoming interfaces, when they all have large bursts coming in at the same time, the MBS will be reduced depending on each incoming burst size. So, the MBS provides the average ceiling of the burst length to the measured path. In our study of TCP behavior on high-speed networks, TCP congestion window steadily agreed with the average MBS over high bandwidth delay product paths. Applications should make their own judgement on how to use this information to avoid dropping packet. In above example, the burst control used one half of the MBS to achieve 500 Mb/s throughput. In FAC^2 algorithm, the maximum packet train length is less than one quarter of the MBS to accommodate the abrupt variation of cross traffic, thus reducing the chance of dropping packet for all traffic.

III. Distinguish and Define Bandwidth and Throughput

It is important to distinguish between bandwidth and throughput.

- **Bandwidth** — the speed that a network element can forward traffic. Both physical and available bandwidths are independent of end hosts and protocol type.
- **Throughput** — amount of data that is successfully sent from a host to another via a network. It is determined by every component along the path from source host to the destination host, including hardware and software. Throughput also has two characteristics — achievable and maximum.

Confusions often occur between available bandwidth and achievable throughput, and between capacity and maximum throughput. Some people think if they maximize the throughput via some tuning techniques, this throughput is equal to the available bandwidth, and some people think that the maximum UDP throughput represents the bandwidth close to the capacity. The MBS theory proves that these thoughts are incorrect.

To illustrate this further, assume that there is a 10 Gb/s network between site A and site B; site A uses a PC hardware for the source host (to send measurement traffic), and site B has a receiver host equipped with the fastest CPU and network adapter available. We would like to answer two questions:

(1) If the available bandwidth from A to B is 9.5 Gb/s, what is the maximum throughput that a TCP application can achieve from A to B?

(2) If the available bandwidth from A to B is 2 Gb/s, what is the maximum throughput that a UDP application can achieve from A to B?

Obviously, these questions cannot be answered with the given information, as they depend on many other factors, such as transmission method, traffic type, transmission host and others. Each of these are explained in more detail below.

1) Maximum Burst Size (MBS)

As discussed in previous section.

2) Transmission Host

If the transmission host uses up-to-date PC system equipped a 64-bit/133MHz PCI-X I/O bus, the maximum throughput can be up to 8 Gb/s depending on the system memory bandwidth and other factors.

3) Protocol Type

Regardless of the current traffic (protocol) type for scenario (1), the achievable throughput of a TCP application can vary from less than 1 Mb/s to approximately 8 Gb/s. Achievable throughput will never be close to the available bandwidth (9.5 Gb/s), no matter what performance tuning techniques [7] are used, due to above PC hardware limitation. In scenario (2), if 50 percent of current traffic is TCP, and a newly launched UDP application runs long enough, this UDP application may achieve more than 5 Gb/s throughput by forcing TCP traffic back off, even though the current available bandwidth is only 2 Gb/s.

4) Other Factors

TCP performance will be affected by operating system, TCP implementation, end-to-end round trip time, receiving and transmitting buffers' size, and so on. Many other factors, such as CPU, PCI chipset, interrupt frequency, and context switch time, all can affect application's achievable throughput.

Therefore, application's achievable throughput is not necessarily determined by available bandwidth. The achievable throughput is the characteristic for applications to base their end-to-end performance expectations, and that available bandwidth is computed as capacity minus the current traffic.

Based on the illustrations above, bandwidths are defined as:

- **Capacity** (*C*) is the maximum number of bits per second a network element can transfer. The capacity of an end-to-end path is determined by the slowest network element along the path.

- **Utilization** (U) is the percentage of the capacity on a link or path currently being consumed by aggregated traffic.

$$U = \frac{Traffic}{C}$$

- **Available bandwidth** (A) is the capacity minus cross traffic (utilization) over a given time interval. This is applicable to paths, links, or routers and switches.

$$\begin{aligned} A(t_s, t_e) &= \text{Capacity - Traffic} \\ &= C \times (1 - U) \\ &\neq A(T_{window}) \end{aligned}$$

$T_{window} = t_s - t_e$

t_s is the time when the measurement started

t_e is the time when the measurement ended

Sampling interval — Measurements of network available bandwidth depend strongly on the time interval used for the measurement. For example, Fig. 4 shows the network available bandwidth measured with both 1 ms and 2.5s time intervals, and shows that the available bandwidth measurement varies greatly depending on what time interval is used. 2.5s samples are never greater than 12% (representing relatively average value) while 1 ms samples peak at 43% (showing instantaneous measurement, compare to 2.5s). The accuracy requirement for available bandwidth depends on the measurement interval. For short interval measurement, it needs to be accurate to reflect utilization at that moment. For a long period measurement, the available bandwidth should have two values, an average value and a range between the minimum value and the maximum value.

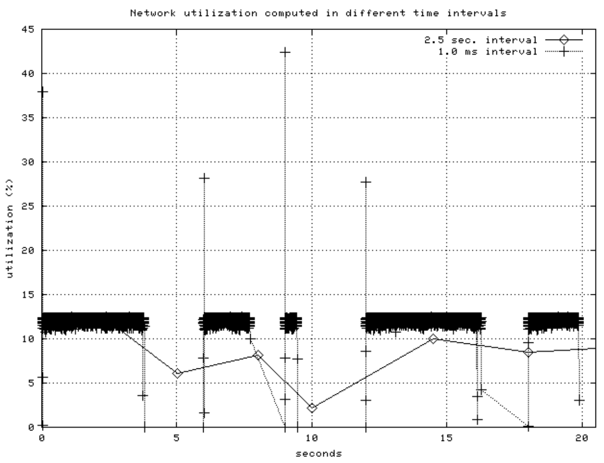


Fig. 4 Available bandwidth at 2 different time intervals

Note that all bandwidth characteristics are at the physical layer, and are independent of the higher-layer protocol (e.g., TCP or UDP)

IV. Existing Algorithms

This section analyzes several algorithms and their mathematical models for closely measuring network bandwidths, and describes one new algorithm — fluid spray effect (FSE), an alternative algorithm to packet pair dispersion, for hop-by-hop bandwidth measurement. The word, “estimation” or “estimate”, means that the measurement result is based on a probe’s approximation, not based on a formula.

VPS algorithms (SD and HD):

Pathchar is a tool for estimating links’ capacity. *Variable packet size* (VPS) algorithm includes size differential (SD) and *hop differential* (HD)[8] methods. The SD algorithm measures the time difference, ΔT, for a constant ΔS, the size difference for packet size increment, by sending UDP packets from the source host to each network element and measuring the time while getting ICMP response (Fig. 5), thus transfer rate can be denoted as:

$$R_{Tx} = \frac{\Delta S}{\Delta T}$$

$\Delta S = S_2 - S_1$, S_1 and S_2 are sizes for two different packets

$\Delta T = T_2 - T_1$, T_1 and T_2 are the time to send packets S_1 and S_2 to a router respectively.

This algorithm has a restriction on the maximum ΔT that depends on the maximum packet-size difference, which is limited by the MTU (Maximum Transfer Unit) of the network interface. For example, if the network

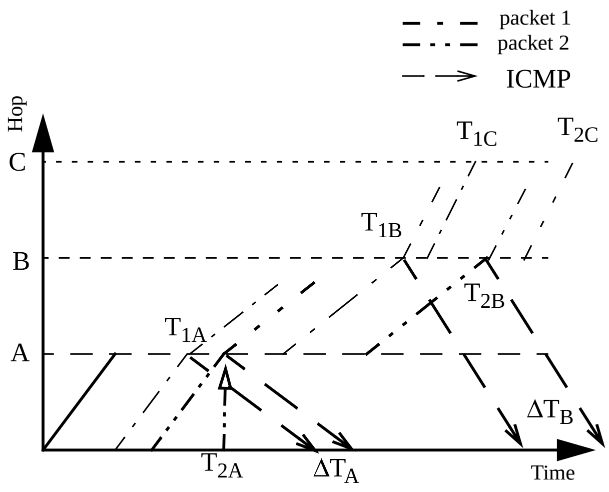


Fig. 5 VPS transfer timing of two packets on a network route

interface is ethernet the maximum size difference ΔS is 1472 Bytes. When a link bandwidth (BW) is OC-3 or better, the ΔT will be smaller than $1472 \times 8 \div 155 \bullet 10^6 = 75.974\ \mu s$. A typical non-local round-trip time (RTT) is greater than 1ms, and typical computer system and RTT fluctuation cause ±5% error rate in time measurement, so the deviation of RTT (Δ_{RTT}), is greater than 50µs. Under these circumstance, the time difference becomes

$$\begin{aligned}\Delta T &= T_l - T_s \\ &= (S_l \div BW + RTT_l) - (S_s \div BW + RTT_s) \\ &= \Delta T_{(\text{zero traffic})} \pm \Delta_{RTT} \qquad (1)\end{aligned}$$

where

S_l and S_s are sizes of largest and smallest packets

T_l and T_s are the time to transfer each of these two packets

$RTT = T_{sys} + T_{ps} + T_q + T_{ack}$

T_{sys} is the system call time

T_{ps} is the time to send (copy) a packet from user space to the edge of a network interface card (NIC) or the reverse.

Tq is the queueing time for both directions

Tack is the time for acknowledgment to travel back

That is, transmission is not linear to packet size in the real network.

The time difference between the largest packet and the smallest packet that can be transmitted from a source host to an intermediate router, is inaccurate when Δ_{RTT} has a magnitude similar to $\Delta T_{(\text{zero traffic})}$, and thus dominates ΔT. So, this algorithm is only good for probing networks with capacity up to OC-3 (155 Mb/s) when the MTU is 1500 bytes. In a network where jumbo frame is used, this algorithm may measure capacity up to 1 Gb/s.

Since cross traffic can cause T_1 and T_2 to vary (non linear), a single probe will not get accurate result. In order to obtain a more accurate result, this algorithm sends a numbers of different size packets to measure the bit rate for each packet, and then uses linear regression to converge the result. The main merit of this algorithm is that the source host does not need high-transfer-rate hardware to measure bandwidth on high-speed networks.

Fig. 5 is the VPS timeline for transferring two packets. It shows that R_{Tx} on first hop represents the link capacity, and R_{Tx} on the remaining hops does not because of store and forward delay. To acquire the time difference between router N and router N+1, hop differential (HD) is needed.

In Fig. 5, the time axis has been nudged at source host so that start time of both packet 1 and 2 is aligned at the time 0 on this graph. At hop 1 (source host to router A), these packets leave router A at different time due to the store and forward delay. This means that $\Delta T_B = T_{2B} - T_{1B}$ does not represent the time difference of transferring these two packets from A to B. Fig. 5 shows that the store and forward delay between these two packets at router A is $\Delta T_A = T_{2A} - T_{1A}$. So, the real time difference between transferring these two packets from A to B is:

$$\Delta T_{AB} = \Delta T_B - \Delta T_A = T_{2B} - T_{1B} - (T_{2A} - T_{1A})$$

and the bandwidth of this link is:

$$BW = \Delta S \div \Delta T_{AB}$$

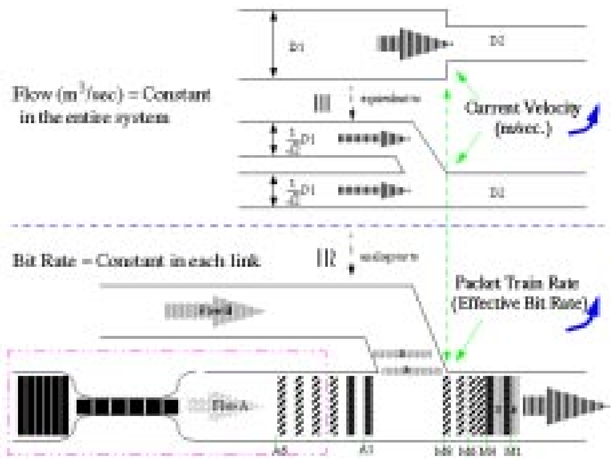
This is the hop differential algorithm. Unfortunately, this algorithm has two fatal issues. First, different routers may have different ICMP response time. This creates difficulties for algorithms based on the hop differential calculation. The reason that pathchar sometimes gives negative results is due to this issue. Second, if any of network element has no ICMP response (called hidden device) is immediately prior to the measured router, the hop differential algorithm will result in a lower bandwidth, which can be computed according electronic capacitor serialization formula:

$$BW = \frac{BW_A \times BW_B}{BW_A + BW_B}$$

BW_A and BW_B are physical bandwidths of router A and B. Although HD algorithm fails to measure the bandwidth when hidden switch exists, it can be used to detect hidden devices.

PPD algorithm:

Packet pair dispersion (PPD) is used in *nettimer* to analyze the bottleneck link capacity. This algorithm is demonstrated in the dot line box at the lower left corner in Fig. 6. The PPD algorithm says that *if a pair of packet travels back-to-back through a bottleneck link, the last bits of two packets are further separated. After they leave the bottleneck link, this separation will remain till the destination. So, the packet pair dispersion represents the narrow (bottleneck) link's capacity.* This is true if and only if no cross traffic happens at the later links. The Internet almost always has traffic, which will cause the fluid spray effect (FSE) when many traffic streams come in from different interface and routed out at another interface, and all packets will bunched together, so the packet pair dispersion theory will not apply. However, using ICMP message to feed this information back to the source host can detect where the bottleneck is.



D1
D2
Flow (m³/sec) = Constant in the entire system
Current Velocity (m/sec.)
D2
Bit Rate = Constant in each link
Packet Train Rate (Effective Bit Rate)
A1

Fig. 6 Packet pair and Fluid Spray Effect (FSE)

FSE theorem: Assume two packet trains, both train rates are lower than the line speed, encounter each other at a router. If aggregated rate equals or exceeds the router's capacity, all packets are bunched together to form a new stream. When this stream leaves the router, its train rate is the outgoing interface (line) speed. See lower right in Fig. 6.

FSE happens almost everywhere on the Internet. Since the bunchy extent is different at each router due to each router's bandwidth and traffic flow, it does not deliver useful information for traffic from a source host to a destination host across a network. However, if the bunchy extent can be fed back to the source host via ICMP message, the ICMP message will carry each router's information back to the source host, and the source host can use it to compute each router's physical bandwidth. Fig. 6 shows how if an incoming train is long enough, a pair of packets or a sub train within this train will have train rate at the line speed when it leaves the router. This is the method used for packet train technology to measure capacity beyond a narrow link in NCS.

OWD algorithm:

Pathload uses packet trains to detect *one-way delay* (OWD)[6] to measure available bandwidth. Theoretically, this algorithm may closely measure the available bandwidth using bisection method. The actual measurement result will vary especially when measuring a high-speed network due to the hardware capability and implementation. As mentioned in § II. , to use packet trains to measure bandwidth, both sender and receiver hosts must have higher hardware bandwidth than the network available bandwidth. To cause OWD, the probe stream must have a higher transfer rate (R_{snd}) than the available bandwidth (A_{bw}). The amount of the difference between R_{snd} and A_{bw} depends on the path capacity (C_p). Intuitively, this difference is directly proportional to the resolution of the system timer. The higher the resolution of the system timer, the smaller the difference required to determine the OWD; thus the result is more accurate. The minimum time needed to distinguish the delay can be either a fixed amount of time or some percentage of the time needed to finish the burst transfer. If the receiver has up-to-date PC hardware with 1 µs time resolution and an on NIC timestamp timer, a few micro seconds (resolution of getting system time) or the time for two system calls (read data and get system time), whichever is greater, can be the bottom line for the time difference. The basic requirement for this algorithm is that the source host needs to have the higher transfer rate than the available bandwidth.

Pathload uses Pair-wise Comparison Test (PCT) and Pair-wise Difference Test (PDT) metric and statistics to detect the OWD trend. This algorithm builds a region, called the gray area, that can be adjusted to estimate the A_{bw}. Metric results above the gray area represent strong OWD trend, and below the gray area represent no OWD trend. Then, the gray area is the range of estimated available bandwidth. Thus, current pathload implementation is designed to estimate the path available bandwidth.

V. FAC² ALGORITHM

This section introduces FAC² — Feedback Adaptive Control and Feedback Asymptotic Convergence — algorithm, and addresses its accuracy and non intrusiveness.

Using packet train, when the sending rate (R_{snd}) is less than or equal to the available bandwidth (note: this is for a specific time interval), the receiving rate (R_{rcv}) should be equal to the sending rate. When R_{snd} is greater than A_{bw} and the packet train length is less than MBS (to avoid packet loss), according the model addressed in Fig. 2, the receiving rate can be expressed as

$$R_{rcv} = \frac{\sum_{i=2}^{n} PT_i}{\left(\sum_{i=2}^{n} PT_i\right) + \sum_{j=0}^{m} XT_j} \times C_p$$

$$= \frac{PT}{PT + XT} \times C_p = \frac{R_{snd}}{R_{snd} + R_{xt}} \times C_p \qquad (2)$$

From this receiving rate (R_{rcv}), we can compute cross traffic rate (R_{xt}) as

$$R_{xt} = \frac{R_{snd}}{R_{rcv}} \times C_p - R_{snd} \tag{3}$$

Available bandwidth (A_{bw}) is then computed as:

$$\begin{aligned} A_{bw} &= C_p - R_{xt} = C_p - \left(\frac{R_{snd}}{R_{rcv}} \times C_p - R_{snd}\right) \\ &= C_p - R_{snd} \times \left(\frac{C_p}{R_{rcv}} - 1\right) \end{aligned} \tag{4}$$

or

$$= R_{snd} - C_p \times \left(\frac{R_{snd}}{R_{rcv}} - 1\right) \tag{4´}$$

Since the capacity (C_p) is unknown, the next higher network standard capacity C'_p (i.e., 567 Mb/s $\Rightarrow$ OC-12, 789 Mb/s $\Rightarrow$ GigE, ... for building error formula), an estimated capacity based on measured maximum throughput, is used. When using the estimated capacity in equation (4´), the estimated available bandwidth A'_{bw} is:

$$A'_{bw} = R_{snd} - C'_p \times \left(\frac{R_{snd}}{R_{rcv}} - 1\right)$$

and the estimation error of A'_{bw} is

$$error = (C'_p - C_p) \times \left(\frac{R_{snd}}{R_{rcv}} - 1\right) \tag{5}$$

In equation (5), even through error is introduced by C'_p, the R_{snd} to R_{rcv} ratio (S/R ratio) has greater effect. If this ratio is 1, that is, the R_{snd} is equal to the R_{rcv}, then the error will be zero. This means when replacing new sending rate (R_{snd}) with the current receiving rate (R_{rcv}) in equation (4), we will obtain a new R_{rcv} that is close to the A_{bw}:

$$\begin{aligned} A_{bw} &= C_p - \frac{R_{snd}}{R_{snd} + R_{xt}} \times C_p \times \left(\frac{C_p}{R_{rcv}} - 1\right) \\ &= C_p - \frac{R_{snd}}{R_{snd} + \left(\frac{R_{snd}}{R_{rcv}} \times C_p - R_{snd}\right)} \times C_p \times \left(\frac{C_p}{R_{rcv}} - 1\right) \\ &= C_p - R_{rcv} \times \left(\frac{C_p}{R_{rcv}} - 1\right) \\ &= C_p - (C_p - R_{rcv}) = R_{rcv} \end{aligned}$$

This new equation expresses that the final receiving rate will be the available bandwidth after repeatedly replacing the R_{snd} with new R_{rcv}, and the error is minimized. This is an asymptotic formula that converges very quickly. The S/R ratio minus 1 is the converging coefficient, and the converging function can be obtained from equation (2) when replacing R_{snd} with current R_{rcv}:

$$R_{rcv0} = \frac{R_{snd0}}{R_{snd0} + R_{xt}} \times C_p$$

$$\begin{aligned} R_{rcv1} &= \frac{\frac{R_{snd0}}{R_{snd0} + R_{xt}} \times C_p^{\,2}}{\frac{R_{snd0}}{R_{snd0} + R_{xt}} \times C_p + R_{xt}} \times C_p \\ &= \frac{R_{snd0} \times C_p^{\,2}}{R_{snd0} \times (C_p + R_{xt}) + R_{xt}^{\,2}} \end{aligned}$$

...

$$R_{rcvn} = \frac{R_{snd0} \times C_p^{\,2n}}{R_{snd0} \times (C_p + R_{xt})(\ldots)(C_p^{\,n} + R_{xt}^{\,n}) + R_{xt}^{\,2n}} \tag{6}$$

n is the number of iterations for convergence.

In an empty network (R_{xt} = 0), the equation (6) becomes:

$$\begin{aligned} R_{rcvn} &= \frac{R_{snd0} \times C_p^{\,2n}}{R_{snd0} \times (C_p)(\ldots)(C_p^{\,n})} = \frac{C_p^{\,2n}}{C_p^{\,n\frac{(n+1)}{2}}} \\ &= C_p^{\,2n - n\frac{(n+1)}{2}} \end{aligned}$$

In this equation, n is either 1 or 2, and the R_{rcvn} equals to C_p. That is, when the initial sending rate (R_{snd0}) is greater than equal to the bottleneck capacity, the initial receiving rate (R_{rcv0}) equals to the bottleneck speed. When replacing R_{snd} with R_{rcv0}, the new receiving rate (R_{rcv1}) will remain at R_{rcv0}. The number of iterations (n) for this particular case is two. If the R_{rcv1} is less than the R_{rcv0}, it means that A_{bw} is less than C_p, and further probe is needed. If the utilization is less than 50%, five iterations can reduce the error less than 1%. This can be proved using equation (2). The higher the utilization, the more iterations needs to converge. When utilization is close to 100%, the iteration is toward infinite. That is, when R_{xt} = C_p, the equation (6) becomes:

$$R_{rcvn} = \frac{R_{snd0} \times C_p^{\,2n}}{R_{snd0} \times 2 \times (C_p)(\ldots)(C_p^{\,n}) + C_p^{\,2n}}$$

$$= \frac{R_{snd0} \times C_p^{2n}}{R_{snd0} \times 2 \times C_p^{n\frac{(n+1)}{2}} + C_p^{2n}} = \frac{R_{snd0}}{R_{snd0} \times 2 \times C_p^{\frac{n(n-3)}{2}} + 1}$$

When n $\Rightarrow \infty$, $R_{rcvn} \Rightarrow 0$. This, however, is not a problem because the higher the network utilization (higher the R_{xt}), the higher the initial S/R ratio (sending to receiving ratio) will be. If the path utilization is high, the R_{rcv0} will be much lower than the R_{snd0}. When aggregated cross traffic is at line speed of the bottleneck link, the S/R ratio will be two if the R_{snd0} is also at that link speed. This is the upper bound of the S/R ratio. Any S/R value at 2 or above means the path is fully utilized, thus, $A_{bw} = 0$, and the number of iterations is one. The lower boundary of the S/R ratio is 1, the asymptotic base line, which means $R_{snd} \Rightarrow R_{rcv} \Rightarrow A_{bw}$. The initial S/R ratio indicates the A_{bw} range and the S/R ratio is useful to detect cross traffic change. In the implementation, after three iterations ($A_{bw} \neq C_p$), bandwidth and cross traffic can be pre estimated via FAC^2 formula (7) and (3). When the utilization is high (initial S/R ratio close to 2), R_{snd} can be reset to a point close to the available bandwidth according to the pre estimation for further probing, which will reduce the number of iterations for faster convergence. For example, a path had average 159.5 ms round trip time (RTT), an OC-12 (622 Mb/s) bottleneck link, and 90% to 95% utilization on the bottleneck link. With normal converging operation, FAC^2 took 30 seconds to reach 99.4% accuracy; with pre estimation and R_{snd} reset, FAC^2 obtained similar accuracy within 5 seconds.

Because the length of packet train for FAC^2 probe is chosen much less than the measured network pipe, the time for each probe is determined by the round trip time (RTT). Five iterations can be done in less than one second for a RTT shorter than 200 ms. Compare this with pathchar, pathload, and even SNMP, FAC^2 uses very short time and less bandwidth, so it is very less intrusive to both end hosts and the measured network. If applications would like to know how A_{bw} varies over time, periodic measurements can be made. For example, an application might take a measurement every N second. Two types of results are obtained for multiple measurements — the average A_{bw} and a range of A_{bw} (from minimum A_{bw} to maximum A_{bw}).

After obtaining available bandwidth, physical bandwidth can be then directly computed from a set of iterations by formula (7) via (3):

Notice that the initial sending rate, R_{snd0}, cannot be used if pre bandwidth estimation is not performed because the R_{snd0} may be far above the capacity.

$$R_{xt} = \frac{R_{snd2}}{R_{rcv2}} \times C_p - R_{snd2} = \frac{R_{snd1}}{R_{rcv1}} \times C_p - R_{snd1}$$

$$C_p = \frac{R_{snd1} - R_{snd2}}{\frac{R_{snd1}}{R_{rcv1}} - \frac{R_{snd2}}{R_{rcv2}}} \qquad (7)$$

Inference of equations (2) through (5) has proved that the FAC^2 algorithm is able to measure (not estimate) network available bandwidth accurately on a given path. The capacity then can be calculated by equation (7). In mathematics, to obtain A_{bw} is straight forward. In practice, beside the sending rate, the other critical issue is not to disturb current network traffic. To ensure this, the packet train length must be far less than the MBS. In hop-by-hop measurements, when a packet train passes a narrow link, its train rate is reduced (dispersion) and this causes measurement error on further links. One solution is to relay on the FSE algorithm because cross traffic almost always exists.

The basic FSE principle has been illustrated in section IV. To measure the dispersion between two packets traveling over the high-speed networks requires high-resolution timer. For example, a 1500-byte packet travelling through an 1 Gb/s link takes 12 µs, so even a 1-µs timing error will cause 8.33% measurement error. Therefore, FSE requires various methods to implement. One way is to install a special NIC with an on board timestamp timer and to modify the kernel to measure accurate packet pair dispersion. This method is not suitable for average users, so the alternative way is to measure sub train dispersion. Since these topics need numerous paragraphs, they are addressed in a separated paper.

VI. Fundamental Requirements for Measuring Bandwidth

This section addresses fundamental requirements, including hardware and implementation issues, for measuring bandwidth.

Single packet and packet train are two techniques to probe a network for measuring bandwidth. In order to measure high-speed networks, the single packet method requires a high resolution timer due to packet size restraint. The packet train technique has no size restriction, therefore, the time resolution is not crucial. However, it requires that the source host must have a higher sending rate than the available bandwidth, and control the burst size and sending rate. The high sending rate may sound trivial since modern CPU and NIC are fast. In fact, this is more complicated.

In the 1990's, network capacity was the limiting factor in throughput. The end host is now the main factor that limits an application's throughput. A host's memory, I/O bus, network interface card (NIC), or operating system all affect the throughput. Thus, to determine if the end hosts are able to measure the available bandwidth is the first task.

In the past 10 years, network speed has increased by a factor of 1000; CPU clock speed has increased by more than a factor of 30; memory clock speed has increased almost a factor of 20 times. Memory bandwidth, however, has increased by only a factor of 10, and PCI I/O bus bandwidth has increased only a factor of 8. If these growth rates continue for the next decade, the end host will certainly be the throughput bottleneck for network applications.

The main bottleneck in current systems is at the memory and I/O sub system. Fig. 7 shows the data path for sending data from user memory to the NIC. For a system equipped with a 32-bit/33MHz PCI bus, if the memory bus is 100 MHz, the total time needed to transfer data from a user buffer to the NIC is 5 memory cycles: the two fixed cycles plus three memory cycles per bus cycle (100/33). However if the memory bus is 133 MHz, then 6 cycles are required (2+133/33). For example, on an Intel BX chipset based motherboard (33 MHz PCI bus), changing the memory from 100 MHz to 133 MHz increases the memory bandwidth from 190 MB/s to 250 MB/s, but also adds an extra 20 percent transfer time (from 5 cycles to 6 cycles) to move data from user memory to NIC memory. The total gain of throughput is therefore only:

$$\frac{250 \div 6 - 190 \div 5}{190 \div 5} \times 100\% = 9.6\%$$

Simply increasing the memory clock speed does not necessarily result in an equivalent increase in the data transfer rate from user space to the NIC. So, to achieve fast sending rate is not trivial on current and future hardware to

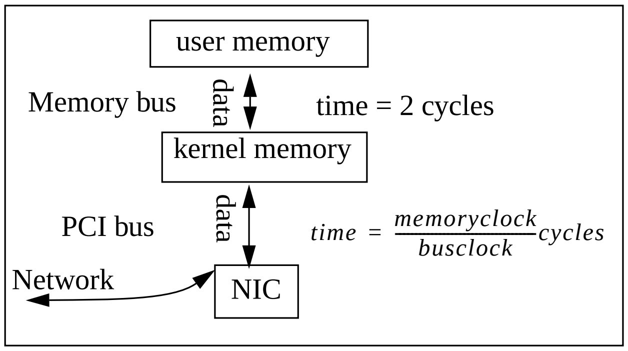


Fig. 7 Hardware data path for packets

compete with network speed. Therefore, as part of the algorithm, we need to measure the host hardware, and analyze its memory bandwidth, CPU power, and the input/output (I/O) bus bandwidth, then compare them with the network interface card (NIC) speed to find the upper limitation of the transfer rate between the hosts at two ends of the measured path. This allows algorithms to determine if these hosts are capable of measuring bandwidths because to measure bandwidth, both hosts must be able to handle data transfer rates higher than the available bandwidth. In other words, the source host must send data fast enough to cause the bottleneck network element to queue packets, and receiving host must be able to handle all incoming data without dropping any packets. Once the system throughput abilities are determined, the source host will send burst to the receiver to measure the receiving rate (R_{rcv}) to estimate the MBS and to determine what the system can do. If the receiving rate is below the maximum throughput of both end hosts, the network available bandwidth is measurable. Otherwise, only the maximum throughput can be measured by these two end systems.

After detecting the MBS, the initial sending/receiving ratio is also obtained. The maximum packet train length then can be determined according these factors. In practice, the cross traffic varies randomly. That is, as available bandwidth changes in time, both MBS and the number of iterations for convergence are affected. However, the major affect of cross traffic variation is on the MBS. When MBS deceasing, the packet loss may occur if burst size (packet train length) is not reduced correspondingly. To accommodate abrupt traffic variations to avoid packet loss, FAC^2 has the maximum train length less than one quarter of the MBS obtained at the begin of the measurement. The real train length will be as short as possible, but the minimum length is determined by accuracy requirement and the initial receiving rate — R_{rcv0}. The higher the accuracy requirement and the R_{rcv0}, the longer the train is needed. The typical range of train length is between 12 and 200 MTU. Also, the total number of iteration for convergence may need to be adjusted when cross traffic varies. This depends on how accurate of the result applications need. Since FAC^2 uses a short packet train, each probe time is determined by the RTT. The small RTT has less range of available bandwidth variation, which can keep the number of iterations required for convergence low.

Application implementation is another important factor affecting the data transfer rate. The implementation also depends on use of which operating system (OS). For example, assume that a system has 1000 MB/s memory bandwidth, and one system call costs 1 µs. Sending a

20KB UDP datagram from user memory to NIC memory takes 100 µs + 1 µs. If this datagram is sent as 20 1KB datagram, then, the total time will be 100 µs + 20 µs. The second method reduces approximately 20% transfer rate. So, in algorithm implementation and code design, both hardware and software issues must take into account.

Rate control has less impact on the A_{bw} measurement. After obtaining the initial receiving rate (R_{rcv0}), the new R_{snd} needs not to be exactly as same as the R_{rcv0} because the initial sending rate (R_{snd0}) may start from any speed, and the R_{rcv0} is not expected to have a fixed value. Therefore, the sending rate (R_{snd}) accuracy does not affect the A_{bw} measurement as long as the R_{snd} is not below the A_{bw}. It, however, may affect the capacity computation when the R_{snd0} is too close to the A_{bw}. Because when R_{snd} is close to the A_{bw} (probe speed close to the asymptotic base line), the S/R ratio will be close to 1, and it is sensitive to the measurement error, thus affecting the capacity computation. Controlling the sending rate will be another whole new topic for implementation, which is not part of this paper because the focus of this paper is on the model and algorithms.

VII. Conclusion

This paper has introduced algorithms — FAC^2 and FSE — for measuring bandwidths, and addressed their requirement and limitations. It has mathematically proved that FAC^2 can produce accurate results of measuring available and physical bandwidth. FAC^2 minimizes the intrusion, and has no impact to current traffic and testing hosts. The limitation of this algorithm is that it cannot measure bandwidths of network elements beyond a bottleneck node in hop-by-hop measurement due to the packet train characteristics. FSE is the algorithm to measure the physical bandwidth beyond the bottleneck link. Therefore, a network bandwidth measurement system can be built from the combination of the FAC^2 and FSE algorithms. Because FAC^2 measures available bandwidth very quickly, consumes very low bandwidth, and does not require any privilege to obtain bandwidth information from routers, ordinary users can use it to monitor the available bandwidth periodically, and easily build MRTG like graphs to visualize and analyze the history of the monitored path. This paper introduced the maximum burst size (MBS) and discussed its critical role in improving network throughput, reducing congestion, and measuring bandwidths.

This paper has described VPS algorithm that has a excellent feature — it can work on any NIC speed to estimate bandwidth of high-speed networks. The paper also analyzed a couple of other algorithms. These algorithms addresses in this paper show a key point that building a mathematical model is a fundamental step to lead success.

VIII. Epilogue

Along with bandwidth measurement development, MBS and FAC2 are also found to be useful for building robust network transfer protocol. During research and development, the MBS based transmission control algorithm has better adaptation to the cross traffic variation than the congestion window based pacing mechanism, which is currently used by transport control protocol (TCP). The congestion window based transmission pacing algorithm is too dull to adapt the network traffic flow change. This is normally to cause packet loss. Also, once it detects packet loss, TCP will reduce the congestion window to one half, which is either insufficient according to MBS theory if a hugh amount of cross traffic comes in, or is unnecessary if there was a short and high-speed burst. Whereas, the transmission pacing mechanism based on MBS can measure the actual queue size and available bandwidth on the bottleneck router, as well as the current cross traffic, thus calculating the current effective queue size — MBS. Then, this pacing algorithm can quickly adjust the sending burst size and pace to adapt the network bandwidth changes. Therefore, MBS based transmission control protocol can efficiently avoid packet loss to fully utilize the available bandwidth.

Other important issue for network bandwidth measurement is the change of the relation between the available network bandwidth and the bandwidth required by end-to-end transmission. Network bandwidth has exceeded the host NIC and system I/O bandwidth, and will continue in this trend in the future. Two problems come up for how to measure high-speed network. (1) Can a slow end host measure a network bandwidth that is higher than host NIC bandwidth and/or I/O system of the end hosts? According current study, existing algorithms are only able to measure the physical bandwidth in such environment. Current algorithms for measuring available bandwidth require that end host has higher throughput than the available bandwidth, and this situation will not be able to measure available bandwidth in the future if the end host I/O is slower than the network bandwidth. So, new algorithm is needed. (2) The resolution of the system timer is another issue for measuring bandwidth. When the bit rate is high, the time for transmit/receive a packet becomes short. The higher the network bit rate, the short of the packet I/O time will be. A 1514-byte packet travelling over an 10Gb/s network takes about 1.21 μs, and this packet travelling over an 1Tb/s network takes only 12.1 ns. Current Unix timer resolution is 1 μs, which is impossible to measure any incoming packet over 5 Gb/s, especially when the receiving interrupt is coalesced, which is almost guaranteed in these high-speed NIC drivers. Therefore, the packet pair dispersion based algorithms are not able to measure bandwidth on the network whose speed is higher than 1Gb/s.

IX. Acknowledgments

I would like to thank Brian Tierney (LBNL) for suggestions on implementation, and literature contribution to this paper. I also thank Jiri Navratil (SLAC) and Margaret Murray (CAIDA) for intensively testing and verifying results during the development.

This work was supported by the Director, Office of Science. Office of Advanced Scientific Computing Research. Mathematical, Information, and Computational Sciences Division under U.S. Department of Energy Contract No. DE-AC03-76SF00098. This is report no. LBNL-48330.

References

[1] Uyless Black, Network management standards: SNMP, CMIP, TMN, MIBs, and object libraries. New York: McGraw-Hill, c1995.

[2] R.L. Carter and M.E.Crovella, “Measuring Bottleneck Link Speed in Packet-Switched Networks,” Performance Evaluation, vol. 27,28, pp. 297-318,1996.

[3] Kevin Lai and Mary Baker. Measuring Bandwidth. In Proceedings of IEEE INFOCOM, March 1999.

[4] Allen B. Downey, Using pathchar to estimate Internet link characteristics, proceedings of SIGCOMM 1999, Cambridge, MA, September 1999, 241-250.

[5] Kevin Lai and Mary Baker, "Nettimer: A Tool for Measuring Bottleneck Link Bandwidth", Proceedings of the USENIX Symposium on Internet Technologies and Systems, March 2001.

[6] C. Dovrolis, P. Ramanathan, D. Moore, What do packet dispersion techniques measure? In Proceeding of IEEE INFOCOM, April, 2001.

[7] Thomas J. Hacker, Brian D. Athey, The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network, Aug. 2001.

[8] G. Jin, G. Yang, B. Crowley, D. Agarwal, Network Characterization Service (NCS), HPDC-10 Symposium, August 2001

[9] Stefan Saroiu, SProbe: A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments. Available: http://sprobe.cs.washington.edu

[10] Manish Jain and C. Dovrolis, Pathload: A Measurement Tool for End-to-end Available Bandwidth, PAM, March, 2002.

[11] Attila Pásztor, Darryl Veitch, Active Probing using Packet Quartets, IMW, Nov. 2002

[12] Deb Agarwal, José María González, Guojun Jin, Brian Tierney, “An Infrastructure for Passive Network Monitoring of Application Data Streams”, PAM, April 2003

[13] G Jin, B Tierney, “Netest: A Tool to Measure the Maximum Burst Size, Available Bandwidth and Achievable Throughput”, ITRE, August 2003

[14] ftp://ftp.arl.mil/pub/ttcp

[15] http://dast.nlanr.net/Projects/Iperf

[16] Netperf: A Network Performance Benchmark. Available: http://www.netperf.org/netperf/training/Netperf.html

[17] pchar: A Tool for Measuring Internet Path Characteristics. Available: http://www.employees.org/~bmah/Software/pchar

[18] http://www.psc.edu/networking/treno_info.html